Application

for

United States Letters Patent

To all whom it may concern:

Be it known that,

Paul A. GASSOWAY

have invented certain new and useful improvements in

METHODS AND SYSTEMS FOR COMPUTER SECURITY

of which the following is a full, clear and exact description:

METHODS AND SYSTEMS FOR COMPUTER SECURITY

BACKGROUND

1. TECHNICAL FIELD

The present disclosure relates generally to security and, more particularly, to methods and systems for computer security.

2. DESCRIPTION OF THE RELATED ART

With the growth of the Internet, the increased use of computers and the exchange of information between individual users poses a threat to the security of computers. Computer security attempts to ensure the reliable operation of networking and computing resources and attempts to protect information on the computer or network from unauthorized corruption, access or disclosure. Computer system(s) as referred to herein may include(s) individual computers, servers, computing resources, networks, etc., and combinations thereof.

Among the various security threats that present increasingly difficult challenges to the secure operation of computer systems are computer viruses, worms, Trojan horses, etc. Computer viruses are programs that can infect other programs by modifying them in such a way as to include a copy of themselves. Unlike computer viruses, worms do not need to infect other programs. Worms are independent programs that are capable of reproducing themselves, spreading from machine to machine across network connections, often via email.

A Trojan horse may be an executable program that appears to be desirable but is merely disguised as "friendly" and actually contains harmful code, allowing an attacker to come in through a "back door" and perform malicious actions on the computer system.

Trojans prey on system vulnerabilities and may be extremely destructive, allowing attackers to monitor, administer, and/or perform any action on a computer system that the user can, just as if they were right in front of it. For a Trojan to gain access to the computer system, the user may first be induced to install the Trojan. For example, this may be done through the offering of anything that a user might find desirable via email, instant messengers, or file sharing tools (i.e., free games, movies, system enhancements, etc.). A user may download a Trojan horse program that appears to be a calculator, performing the functions of a simple pocket calculator. When the user launches the infected file, it may appear to be performing calculations and nothing more. However, it may also be performing a number of harmful actions, such as deleting files, stealing passwords, adding files, disrupting system operation, etc. In addition, the Trojan horse may be an e-mail attachment disguised as a document file, readme file, etc. If a user launches the infected file, the Trojan may initiate installation procedures and/or propagation routines.

Trojan horse programs can be introduced to a computer system by initially being planted in software repositories that many people can access, such as software bulletin boards, publicly accessible directories, file-sharing systems, such as the KaZaA network, etc. Users accessing these repositories are then tricked into copying the Trojan horse program into their own computer systems. These users then can further spread the Trojan horse by sharing the infected program with other users, most especially if the program performs a useful function and causes no immediate or obvious damage.

Users may utilize anti-virus programs in order to protect their computer systems from security threats such as Trojan horses. Anti-virus programs operate to protect from the spread of viruses by detecting the virus and isolating or removing the viral code. Examples

of anti-virus software may include activity monitoring programs, scanning programs, and/or integrity checking programs. Activity monitoring programs attempt to prevent the infection of computer systems by searching for "virus-like" activity, such as, attempts to delete a file, or to write to an executable file, and may then attempt to prevent this activity from taking place. Virus scanning programs may contain a list of previously defined virus signatures, containing the binary patterns of a virus, each associated with a virus and scan the various files of a system looking for a match to a particular virus signature. If a virus is detected, the user may be notified and further steps may be taken to rid the system of the malicious code. Integrity checking programs compute a checksum value for all of the uninfected, executable files residing on the computer system and compare the computed checksum values to checksum values generated at a later time to determine if anything has changed in the file. If the checksums match, then the executable file is uninfected. However, if the checksums do not match, then the executable file may possibly be infected and steps may be taken to remove the infected file.

Anti-virus software programs may not provide a computer user with comprehensive protection against Trojans. For example, activity monitoring programs may not adequately prevent Trojan horses because it is hard for them to distinguish between a Trojan horse that, for example, is maliciously deleting a system's file, and a regular program that is supposed to delete a system's file. Virus scanning software may detect viruses present in the system, but it may do nothing to prevent them from infiltrating the system in the first place. The virus scanning software should be continuously updated in order to be effective in detecting new and modified Trojans. This not only proves to be a very tedious and time consuming task for computer users, but also may not happen often enough to provide adequate safeguards against

foreign intrusions. Integrity checking programs not only do not know which viruses they are in fact detecting; but in cases where a file has been legitimately modified, they may also require the user to verify whether or not the detected executable file contains a virus. There is a window of time between when a new attack is released to the public, and when anti-virus products have signatures to detect the attack. During this window of time, the attack is given the opportunity to do its damage. Therefore, just because a user has installed and is running an anti-virus program does not necessarily mean that the user's system is no longer vulnerable to security threats.

SUMMARY

A method for maintaining computer security, comprises providing a database of known good software, opening a file, identifying the file being opened, determining whether an entry exists in the database of known good software for the identified file and performing at least one of allowing and preventing the opening of the file from continuing based on the result of the determination.

A system for maintaining computer security, comprises a database of known good software, a system for opening a file, a system for identifying the file being opened, a system for determining whether an entry exists in the database of known good software for the identified file and a system for performing at least one of allowing and preventing the opening of the file from continuing based on the result of the determination.

A computer recording medium including computer executable code for maintaining computer security, comprises code for providing a database of known good software, code for opening a file, code for identifying the file being opened, code for determining whether an

entry exists in the database of known good software for the identified file and code for performing at least one of allowing and preventing the opening of the file from continuing based on the result of the determination.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the present disclosure and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

Figure 1 shows a block diagram of an exemplary computer system capable of implementing the method and system of the present application;

Figure 2 shows a flow chart of a method for maintaining computer security, according to an embodiment of the present disclosure;

Figures 3A and 3B show a flow chart and schematic diagram respectively illustrating a system and method for maintaining computer security, according to an embodiment of the present disclosure; and

Figure 4 shows a schematic diagram illustrating the functioning of a call hook, according to an embodiment of the present disclosure.

DETAILED DESCRIPTION

The present disclosure provides tools (in the form of methodologies, apparatuses, and systems) for maintaining computer security. The tools may be embodied in one or more computer programs stored on a computer readable medium or program storage device and/or

transmitted via a computer network or other transmission medium.

The following exemplary embodiments are set forth to aid in an understanding of the subject matter of this disclosure, but are not intended, and should not be construed, to limit in any way the claims which follow thereafter. Therefore, while specific terminology is employed for the sake of clarity in describing some exemplary embodiments, the present disclosure is not intended to be limited to the specific terminology so selected, and it is to be understood that each specific element includes all technical equivalents which operate in a similar manner.

The specific embodiments described herein are illustrative, and many variations can be introduced on these embodiments without departing from the spirit of the disclosure or from the scope of the appended claims. Elements and/or features of different illustrative embodiments may be combined with each other and/or substituted for each other within the scope of this disclosure and appended claims.

Software as the term is used herein may include executable instructions (e.g., one or more programs) and/or data that can be stored electronically. An application is a program or group of programs designed for end users and may include systems software and applications software. Virtually all information stored in a computer is stored in a file. There are many different types of files, including data files, text files, program files, directory files, etc. In effect, a file is a collection of instructions and/or data that has a name associated to it, called a file name.

Figure 1 shows an example of a computer system **100** which may implement the method and system of the present disclosure. The system and method of the present disclosure may be implemented in the form of a program running on a computer system, for

example, a mainframe, personal computer (PC), handheld computer, server, etc. The program may be stored on a recording media locally accessible by the computer system, for example, floppy disk, compact disk, hard disk, etc., or may be remote from the computer system and accessible via a hard wired or wireless connection to a network, for example, a local area network, or the Internet.

The computer system **100** can include a central processing unit (CPU) **102**, program and data storage devices **104**, a printer interface **106**, a display unit **108**, a (LAN) local area network data transmission controller **110**, a LAN interface **112**, a network controller **114**, an internal bus **116**, and one or more input devices **118** (for example, a keyboard, mouse etc.). As shown, the system **100** may be connected to a database **120**, via a link **122**.

According to an embodiment of the present disclosure, a list of known good software is maintained. The list may be in the form of one or more databases provided remotely and/or locally on the computer system. When a file is opened, the system identifies the file and checks to determine whether an entry for the file exists in the list of known good software. If an entry exists, the system is allowed to proceed opening the file without interference. For example, if the file is an executable program file, the program is allowed to execute. However, if an entry for the files does not exist in the list, the system can monitor the execution of the program so that if the program attempts to perform a suspect action, such as a change to the operating system registry, settings, and/or change of another executable's file, etc., the user can be prompted before the program is allowed to continue. The user is thus able to prevent the process from doing damage to the system. If a program has been allowed to run on the system for some time, the system can automatically add an entry for the file to the list of known good software.

A more detailed description of a method for maintaining computer security, according to an embodiment of the present disclosure, will be described with reference to Figure 2. A list (e.g., one or more databases) of known good software is provided (Step **S21**). The database may include entries uniquely identifying each piece of software listed in the database. When a file is going to be opened (Step **S22**) it is identified (Step **S23**) and compared with entries in the list of known good software (Step **S24**). Appropriate operations may then be performed on the file (Step **S25**) depending on whether an entry for the file is in the database for known good software. For example, if the file is a program file, if it is determined that the file corresponds to an entry in the database for known good software (Yes, Step S25), the program can be allowed to freely execute (Step S27). If there is no entry in the database for known good software (No, Step S25), the system can perform an appropriate operation on the file (Step S26) which may include monitoring the program for suspicious activities. For example, as will be described in more detail below, one or more operating system call hooks can be placed and used to monitor the program.

According to an embodiment of the present disclosure, when the file is opened the file may be identified by determining a unique value for the file. For example, the unique value may be a hash value generated in accordance with a number of existing methods and technologies, such as one-way hashing techniques (for example, MD5, SHA, etc.), etc.

According to another embodiment of the present disclosure, a database of unfamiliar software may be provided. When a file is being opened, the file is identified. It is then determined whether the file is listed in the database of unfamiliar software. Appropriate operations may then be performed on the file depending on whether the file is listed in the database. For example, if it is determined that the file is listed in the database for unfamiliar

software and the file is a program file, one or more operating system (OS) call hooks can be placed in the program. Several OS calls may be hooked, including but not limited to, updating the registry, opening files, etc. When the call hook occurs, the execution of the program is halted until it is granted permission to proceed. The operation system call hooks will be described in more detail below.

According to another embodiment of the present disclosure, the database of unfamiliar software may include timestamp information indicating, for example, how long an entry for each unfamiliar file has been in the database of unfamiliar software (e.g., a date stamp), the number of times an unfamiliar file has been opened and/or the number of times an unfamiliar piece of software has been executed, etc.

An embodiment of the present disclosure will be described with reference to Figs 3A and 3B. The operating system 300 opens the file (Step **S30**) and device driver **301** reads and identifies the file (Step S31). Once the file is identified, checking device **302** queries the database of known good software 303 (Step S32) and determines if there is a corresponding entry in the database of known good software **303.** If it is determined that there is an entry for the file in the database of known good software 303 (Yes, Step S33), operating system 300 is allowed to continue opening and utilizing the contents of the file (Step S34). For example, if the file contains an executable, the operating system 300 is allowed to let the executable begin. If it is determined that there is no corresponding entry in the database of known good software **303** (No, Step S33), checking device **302** queries the database of unfamiliar software 304 (Step S35) and determines if there is a corresponding entry in the database of unfamiliar software **304**. If an entry is not found (No, Step S36), an appropriate action can be performed (Step S37). For example, a new entry for the file can be made in the

database of unfamiliar software 304. The entry may include information indicating the date the entry was added to the database. If it is determined that there is an entry for the file in the database of unfamiliar software **304 (Yes, Step S36)**, the operating system is allowed to continue opening the file. However, the system monitors it for suspicious activity. For example, if the file contains an executable, when the process starts, one or more operating system call hooks **305** may be placed and the process is allowed to continue. In addition, if an entry was found in the database of unfamiliar software 304, the date stamp for the file entry can be retrieved (Step S38) and compared with the current date (Step S39). If it is determined that the entry has been in the database for unfamiliar software 304 for a sufficient period of time (e.g., a month or more) (Yes, Step S40), then the entry information can be moved from the database of unfamiliar software 304 to the database for known good software 303 (Step S42) and the system is allowed to continue opening and utilizing the contents of the file (Step S34). If the entry has not been in the database for unfamiliar software 304 for a sufficient period of time (No, Step S40), an appropriate action can be performed (Step S37).

In addition, as mentioned above, entries in the database of unfamiliar software may include the number of times the unfamiliar file has been opened and/or the number of times the unfamiliar piece of software has been executed. These values can be compared with baseline values. If the number is (are) greater than the baseline values, the entry information for the file can be moved from the database of unfamiliar software to the database for known good software.

According to various embodiments of the present disclosure, performing an operating system call hook includes notifying a Trojan notification service that the file corresponds to an entry in the database for unfamiliar processes and prompting the user for input about

whether the operating system call should be passed along or should fail. If the operating system call is passed along, then the operating system is allowed to proceed with opening the file.

The OS call hook exists in kernal space and cannot interact with the user. Therefore, according to an embodiment of the present disclosure, a service such as a Trojan notification service may be notified when the call hook occurs. The service may then notify the user via any suitable manner (e.g., email) of the action that is being performed. When the user then decides to allow or not allow the action, the user can notify the Trojan notification service. The Trojan notification service can then return this information to the computer system. In response, the operating system can allow the call to pass along or fail accordingly.

Figure 4 is a schematic diagram illustrating the functioning of a call hook, according to an embodiment of the present disclosure. Performing an operating system call hook **305** involves "hooking" a file's operating system calls **401** (e.g., updating the registry, opening files, etc.) and not allowing the operation to continue until allowed. When the operating system call hook 305 occurs, the Trojan notification service **402** is notified. Trojan notification service 402 then performs a user prompt **403**, prompting the user of the system for input about whether the operating system call should be allowed. Once the user decides whether to allow or not allow the action, the result is passed back through the Trojan notification service **402** to the operating system call hook 305 which then allows the operating system call to proceed or prevents the operating system from performing the hooked call.

Numerous additional modifications and variations of the present disclosure are possible in view of the above-teachings. It is therefore to be understood that within the scope

of the appended claims, the present disclosure may be practiced other than as specifically described herein.